



Adicionando Suporte para GPS

Overview sobre GPS

O Sistema de Posicionamento Global é de responsabilidade do Programa Conjunto Office (JPO) localizado na Divisão Espacial do Comando de Sistemas da Força Aérea dos EUA, Base Aérea de Los Angeles (AFB). Em 1973, o JPO foi dirigido por Departamento de Defesa dos EUA (DoD) para estabelecer, desenvolver, testar, adquirir, e implantar um sistema de posicionamento espacial. O atual Sistema de Navegação com Sistema de Posicionamento Global (GPS) de Tempo e Alcance (NAVSTAR) é o resultado desta diretiva inicial.

O Sistema de Posicionamento Global foi concebido como um sistema que vai desde posições conhecidas de satélites no espaço para posições desconhecidas em terra, mar, no ar e no espaço. Efetivamente, o sinal de satélite é continuamente marcado com seu (próprio) tempo de transmissão para que, quando recebido, o período de trânsito do sinal pode ser medido com um receptor sincronizado. Os objetivos originais do GPS foram a determinação instantânea de posição e velocidade (ou seja, navegação) e a coordenação precisa do tempo (ou seja, transferência de tempo). Uma definição detalhada dada por Wooden (1985) diz:

“O Sistema de Posicionamento Global Navstar (GPS) é um sistema de navegação espacial para todos os climas em desenvolvimento pelo Departamento de Defesa (DoD) para satisfazer os requisitos para as forças militares determinarem com precisão sua posição, velocidade

e tempo em um sistema de referência comum, em qualquer lugar ou perto da Terra em uma base contínua."

Como o DoD é o iniciador do GPS, os principais objetivos eram militares. Mas o Congresso dos EUA, com orientação do Presidente, dirigiu DoD para promover seu uso civil. Isso foi muito acelerado pelo emprego de o Macrômetro para levantamento geodésico. Este instrumento estava em comercialização uso na época os militares ainda estavam testando receptores de navegação para que o

A primeira aplicação produtiva do GPS foi estabelecer dados geodésicos de alta precisão redes.

Como dito anteriormente, o GPS usa pseudodistâncias derivadas da transmissão sinal de satélite. O pseudorange é derivado da medição do curso tempo do sinal (codificado) e multiplicando-o pela sua velocidade ou medindo a fase do sinal. Em ambos os casos, os relógios do receptor e do satélite são empregados. Como esses relógios nunca estão perfeitamente sincronizados, em vez de intervalos verdadeiros são obtidos "pseudoranges" onde o erro de sincronização (denominado como erro de relógio) é levado em consideração.

Consequentemente, cada equação deste tipo compreende quatro incógnitas: as três coordenadas do ponto contidas na faixa verdadeira e o erro do relógio. Desta forma, quatro satélites são necessários para resolver as quatro incógnitas. De fato, o GPS, assume que quatro ou mais satélites estão à vista em qualquer local da Terra 24 horas por dia. A solução se torna mais complicada ao usar a fase medida. Este observável é ambíguo por um número inteiro de comprimentos de onda do sinal para que o modelo para

pseudoranges de fase seja aumentado por um viés inicial, também chamado de ambiguidade inteira.

O sistema global para todos os climas gerenciado pelo JPO consiste em três

segmentos:

- o segmento espacial que consiste em satélites que transmitem sinais,
- o segmento de controle direcionando todo o sistema,
- o segmento de usuário incluindo os vários tipos de receptores.

Trabalhando com GPS no Angular

Nós chamamos de geofencing o uso da tecnologia de GPS e dentro do desenvolvimento Web e mobile temos várias maneiras de conseguir a localização geralmente no Ionic podemos usar o capacitor, cordova, Google Maps API ou até mesmo o Javascript, porém, temos que considerar caso o projeto seja somente mobile é recomendado usar o capacitor/cordova ou o Google Maps API caso deseje criar um mapa na aplicação mobile.

Vamos usar primeiro o nosso projeto Angular criado anteriormente para capturar a localização do usuário, posteriormente vamos usar o Google Maps API e também mostrar um exemplo da documentação do Ionic sobre GPS em capacitor.

No projeto Angular crie um novo componente na pasta pages através do comando abaixo:



ng g component pages/gps

Vamos criar uma função de getLocation() conforme o código abaixo:

```
getLocation() {

  if (navigator.geolocation) {

    navigator.geolocation.getCurrentPosition((position:
GeolocationPosition) => {

      if (position) {

        console.log("Latitude: " + position.coords.latitude +

         "Longitude: " + position.coords.longitude);

        this.lat = position.coords.latitude;

        this.lng = position.coords.longitude;

        console.log(this.lat);

        console.log(this.lat);

      }

    },

      (error: any) => console.log(error));

  } else {
```

```
    alert("Geo :");
  }
```



Agora vamos definir algumas variáveis no componente e adicionarmos a função ngOnInite.

```
public lat = 0;

public lng = 0;

public ngOnInit(): void {

  this.getLocation();

}
```

Agora vamos alterar o HTML desse componente.

```
<h1>Angular Web GPS</h1>

<p>

 Exemplo de GPS usando Angular e Javascript<br>

</p>

<h1>

 Segue sua localização<br>

 latitude:-{{lat}}<br>

 longitude-:{{lng}}
```

</h1>



GPS no Ionic Capacitor

Agora, vamos continuar nosso projeto de Ionic vamos usar o Google Maps Api, porém antes vamos dar uma olhada na documentação sobre os serviços de localização do Ionic.

Como podemos verificar na URL: https://capacitorjs.com/docs/apis/geolocation

Primeiro temos que instalar a dependência conforme o comando abaixo:

```
npm install @capacitor/geolocation

npx cap sync
```

Pronto você já tem a dependência instalada agora no seu componente vamos usar o código abaixo:

```
import { Geolocation } from '@capacitor/geolocation';

//temos que criar essa função para conseguir as coordenadas.

const printCurrentPosition = async () => {

  const coordinates = await Geolocation.getCurrentPosition();

  console.log('Current position:', coordinates);

};
```

É necessário fazer algumas alterações de permissão tanto para Android quanto para IOS conforme a documentação inforn , da URL anterior.

IONIC com Capacitor Maps API

Vamos criar um projeto de exemplo para isso podemos usar o mesmo projeto ou criar um novo.

Você deve registrar a chave no Google Maps API conforme o vídeo a seguir

https://www.youtube.com/watch?v=aTYg_pDXHDA

Após registrar a chave da API vamos seguir as instruções da documentação do Capacito através da URL a seguir: https://capacitorjs.com/docs/apis/google-maps

Agora vamos começar a construir nosso componente, vamos gerar um novo ou usar um existente.

Ionic g page map

Agora vamos começar a alterar nosso componente.

```
import { Component, ElementRef, ViewChild } from '@angular/core';

import { Capacitor } from '@capacitor/core';

import { GoogleMap, MapType } from '@capacitor/google-maps';

import { environment } from 'src/environments/environment';
```

Adicionamos a variável do mapa.



```
@ViewChild('map') mapRef: ElementRef<HTMLElement>;

newMap: GoogleMap;

center: any = {

  lat: -23.0008826,

  lng: -43.3505312,

};

markerId: string;
```

Criamos o construtor

```
constructor() {}

ionViewDidEnter() {

  this.createMap();

}

async locate() {

  if(this.newMap) await this.newMap.enableCurrentLocation(true);

}
```

posteriormente uma função locate para pegar a localização do usuário.

Agora vamos instaciar o mapa para ele ser carregado, para isso vamos usar uma função denominada createMap.

```
async createMap() {
  try {
   this.newMap = await GoogleMap.create({
    id: 'capacitor-google-maps',
    element: this.mapRef.nativeElement,
    apiKey: environment.google_maps_api_key,
    config: {
     center: this.center,
     zoom: 13,
    },
   });
   console.log('newmap', this.newMap);
   await this.addMarker(this.center.lat, this.center.lng);
   await this.addListeners();
  } catch(e) {
   console.log(e);
  }
```

```
}
```



Autor: Nykz https://github.com/Nykz/ionic-capacitor-googlemaps-android-ios

Nessa função vamos criar também um marcador e para isso criamos uma função para adicionar o marcador.

```
async addMarker(lat, lng) {
  // Add a marker to the map
  if(this.markerId) this.removeMarker();
  this.markerId = await this.newMap.addMarker({
    coordinate: {
      lat: lat,
      lng: lng,
    },
    title: 'Posto de Gasolina',
    draggable: true,
  });
}
```

Temos que criar as funções de Listerners que vão observar as ações do mapa conforme o código abaixo.

```
async addListeners() {

  // Handle marker click

  await this.newMap.setOnMarkerClickListener((event) => {

   console.log('setOnMarkerClickListener', event);

   this.removeMarker(event.markerId);

  });

  await this.newMap.setOnCameraMoveStartedListener((event) => {

   console.log(event);

  });

  await this.newMap.setOnCameraIdleListener((event) => {

   console.log('idle: ', event);

   // alert(event);

   this.center = {

    lat: event.latitude,

    lng: event.longitude

   };

   this.addMarker(this.center.lat, this.center.lng);
```



```
});

await this.newMap.setOnMapClickListener((event) => {
  console.log('setOnMapClickListener', event);
  this.addMarker(event.latitude, event.longitude);
});

await this.newMap.setOnMyLocationButtonClickListener((event) => {
  console.log('setOnMyLocationButtonClickListener', event);
  this.addMarker(event.latitude, event.longitude);
});

await this.newMap.setOnMyLocationClickListener((event) => {
  console.log('setOnMyLocationClickListener', event);
  this.addMarker(event.latitude, event.longitude);
});
}
```

Durante as vídeo aulas vamos fazer um exemplo prático ao vivo e demonstrar o funcionamento do aplicativo de mapas.

Salvando Coordenadas no Banco de Dados

Esta é uma das formas mais fundamentais de armazena dos geocoordenados.

Os valores de latitude e longitude podem ser representados e armazenados em um banco de dados SQL usando pontos decimais (graus decimais) em vez de graus (ou graus minutos segundos).

Por exemplo, se eu pegar a geolocalização de Colombo que podemos representar como, (Latitude DMS) 6° 55′ 57″ N = 6,932500 (Latitude em Graus Decimais)

(Longitude DMS) 79° 51′ 27,4392″ E = 79,857622 (Longitude de Graus Decimais)

Dependendo da precisão que você pode exigir, você pode decidir quantos pontos decimais você precisa para manter os valores de latitude e longitude. Essa decisão pode levar à escolha de usar float ou decimal no banco de dados SQL.

Uma outra situação que podemos colocar é armazenar até 5 casas decimais que será suficiente para diferenciar duas árvores em um local distante. Portanto, um tipo de dados como DECIMAL (6,5) no MySQL é suficiente neste caso.

Nota: Tenha cuidado ao usar o tipo de dados floats, pois isso pode arredondar o valor.

Tipos de dados espaciais

Tipos espaciais são tipos de dados que armazenam dados de geometria. Cada aspecto ao nosso redor tem algum tipo de componente espacial. Podemos visualizar onde estamos vivendo, de

que forma estamos viajando, etc. Portanto, os mapas são apenas uma das maneiras de utilizar dados espaciais.

A maioria dos bancos de dados, incluindo MySQL, já vem no seu pacote com os tipos de dados espaciais, automaticamente. Entretanto, outros como o Postgres precisam de alguma configuração ou mesmo de uma instalação para usar um tipo de dado espacial.

Vamos considerar o MySQL primeiro, seguindo as especificações OGC (O pen G eospatial C onsortium), o MySQL irá implementar extensões espaciais como um subconjunto do ambiente SQL com diferentes tipos de Geometria. Como resultado, há mudanças drásticas nesses tipos de dados espaciais entre o MySQL 5.6 e 8.0. Para entender o que são, primeiro, você precisa saber o que é SRID .

SRID (Identificador de Referência Espacial)

Pode haver vários sistemas de coordenadas dependendo do plano ou forma. Por exemplo, um plano cartesiano é definido como espaço plano. Um plano geográfico é de um elipsóide. Assim, um determinado ponto na superfície pode ter vários significados diferentes com uma base em qual sistema de coordenadas ele se encontra.

SRID, que significa Identificador de Referência Espacial , descreve o espaço de coordenadas no qual o objeto geométrico é definido. Para MySQL, SRID é um número inteiro único associado a um sistema de coordenadas que o OGC propôs.

Por padrão, o MySQL usa SRID 0, que representa um plano cartesiano plano infinito sem unidades atribuídas a seus eixos. Nas versões 5.7 e anteriores, o SRID foi ignorado e todos os cálculos são cartesianos.

O MySQL 8.0, por outro lado, suporta vários sistemas de referência espacial e cálculos geográficos. Isso significa que SRIDs de geometrias têm significado e afetam os cálculos. Além disso, existem algumas melhorias notáveis em comparação com a versão 5.6, o que é benéfico para o cálculo da distância.

Ele introduz a computação elipsóide em vez do plano cartesiano.

Introduziu vários identificadores de sistemas de referência espacial (SRIDs). Dentre eles, os seguintes são notavelmente benéficos para o nosso caso de uso.

- 4326 — Representa dados espaciais usando coordenadas de longitude e latitude na superfície da Terra, conforme definido no padrão WGS84 , que também é usado para o Sistema de Posicionamento Global (GPS)

- 3857 — Usado por determinados aplicativos de mapeamento e visualização da Web (Google Maps, Bing). Usa desenvolvimento esférico de coordenadas elipsoidais.

Para salvar as informações vai depender muito do seu back-end se utilizar mongo DB você pode armazenar como um double ou numérico.

Atividade Extra



Para se aprofundar no assunto desta aula assista o vídeo de Como O GPS Funciona - Para Fácil Compreensão do INCRÍVEL.

Canal: INCRÍVEL

Título para buscar no youtube: **Como O GPS Funciona - Para Fácil Compreensão**

Referência Bibliográfica

ACKERMANN, F. and SCHDE, H. **Aplication of GPS for Aerial Triangulation. Photogrammetric Engineering & Remote Sensing**. v 59, n 11. – Special Issue: GPS Photogrammetry. november 1993.

DEITEL, Paul; DEITEL, Harvey. **Java como programar**. Bookman, 2005.

GATTASS, Marcelo. **Servlets e COM para a Visualização de Dados Geográficos na Web**, Rio de Janeiro, [2000].

JÚNIOR, Edvaldo Simões da Fonseca. **Global positioning system**. São Paulo, [2003].

MAGRO, F. H. S. **GPS & Aerotriangulation. International Society for Photogrammetry and Remote Sensing-ISPRS**. Washington. 1992.

SILVA, Miguel Reis Castilho da. **Uma introdução sobre sistemas de localização**. Évora, [2003].



Atividade Prática 13 - Adicionando Suporte para GPS

Título da Prática: Criação de um teste de localização pelo celular.

Objetivos: Praticar a programação para informar a localização (coordenadas) de um dispositivo móvel quando entra ou sai de uma área específica.

Materiais, Métodos e Ferramentas: Iremos fazer uma pesquisa na internet e utilizar os conceitos de localização do GPS na programação.

Atividade Prática

O Sistema de Posicionamento Global é de responsabilidade do Programa Conjunto Office (JPO) localizado na Divisão Espacial do Comando de Sistemas da Força Aérea dos EUA, Base Aérea de Los Angeles (AFB). Em 1973, o JPO foi dirigido pelo Departamento de Defesa dos EUA (DoD) para estabelecer, desenvolver, testar, adquirir, e implantar um sistema de posicionamento espacial. O atual Sistema de Navegação com Sistema de Posicionamento Global (GPS) de Tempo e Alcance (NAVSTAR) é o resultado desta diretiva inicial.

O Sistema de Posicionamento Global foi concebido como um sistema que vai desde posições conhecidas de satélites no espaço para posições desconhecidas em terra, mar, no ar e no espaço. Efetivamente, o sinal de satélite é continuamente marcado com seu (próprio) tempo de transmissão para que, quando recebido, o período de trânsito do sinal pode ser medido com um receptor sincronizado. Os objetivos originais do GPS foram a determinação instantânea de posição e velocidade (ou seja, navegação) e a

coordenação precisa do tempo (ou seja, transferência de tempo). Uma definição detalhada dada por Wooden (1985) diz:

"O Sistema de Posicionamento Global Navstar (GPS) é um sistema de navegação espacial para todos os climas em desenvolvimento pelo Departamento de Defesa (DoD) para satisfazer os requisitos para as forças militares determinarem com precisão sua posição, velocidade e tempo em um sistema de referência comum, em qualquer lugar ou perto da Terra em uma base contínua."

Como o DoD é o iniciador do GPS, os principais objetivos eram militares. Mas o Congresso dos EUA, com orientação do Presidente, dirigiu DoD para promover seu uso civil. Isso foi muito acelerado pelo emprego do Macrômetro para levantamento geodésico. Este instrumento estava em comercialização uso na época os militares ainda estavam testando receptores de navegação para que o

A primeira aplicação produtiva do GPS foi estabelecer dados geodésicos de alta precisão redes.

Como dito anteriormente, o GPS usa pseudodistâncias derivadas da transmissão sinal de satélite. O pseudorange é derivado da medição do curso tempo do sinal (codificado) e multiplicando-o pela sua velocidade ou medindo a fase do sinal. Em ambos os casos, os relógios do receptor e do satélite são empregados. Como esses relógios nunca estão perfeitamente sincronizados, em vez de intervalos verdadeiros são obtidos "pseudoranges" onde o erro de sincronização (denominado como erro de relógio) é levado em consideração.

Consequentemente, cada equação deste tipo compreende quatro incógnitas: as três coordenadas do ponto contidas na faixa verdadeira e o erro do relógio. Desta forma, quatro satélites são

necessários para resolver as quatro incógnitas. De fato, o GPS, assume que quatro ou mais satélites estão à vista em qualquer local da Terra 24 horas por dia. A solução se torna mais complicada ao usar a fase medida. Este observável é ambíguo por um número inteiro de comprimentos de onda do sinal para que o modelo para pseudoranges de fase seja aumentado por um viés inicial, também chamado de ambiguidade inteira.

Baseado nos conceitos do GPS, desenvolva uma aplicação para informar apenas a localização (as coordenadas) do GPS.

Gabarito Atividade Prática

Para responder essa questão basta criar um factoryserviço no qual recebemos informações sobre a geolocalização de JSON API. Crie um arquivo js/location.jse escreva:

```
app.factory('myCoordinates', ['$q', '$http', function myCoordinates($q, $http) {

var deferred = $q.defer();

$http.get('http://ip-api.com/json')

        .success(function(coordinates) {

                var myCoordinates = {};

                myCoordinates.lat = coordinates.lat;

                myCoordinates.lng = coordinates.lon;
```

```
            myCoordinates.city = coordinates.city;

            deferred.resolve(myCoordinates);

    })

    return deferred.promise;

    }])
```



Ir para exercício